Carnegie Mellon University

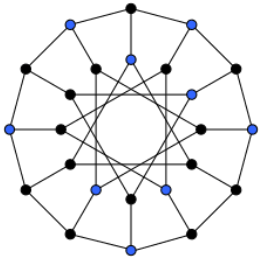
FrontierCO: Real-World and Large-Scale Evaluation of Machine Learning Solvers for Combinatorial Optimization

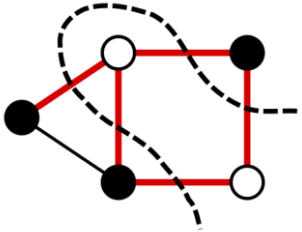
Shengyu Feng, Weiwei Sun*, Shanda Li, Ameet Talwalkar, Yiming Yang*

ICLR

Machine Learning for Combinatorial Optimization

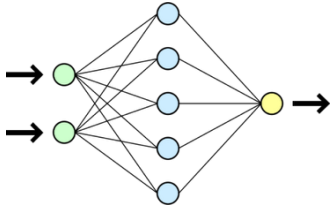
CO Problems





Machine-learning based CO Solvers

ML4CO solvers



Parameterized neural solvers

```
import coptpy as cp
from coptpy import COPT
# Initialize the optimization environment
env = cp.Envr()
model = env.createModel("TransportationOptimization")
# Define costs and capacities for each mode of transportation
costs = {'trucks': 100, 'airplanes': 120, 'ships': 130}
capacities = {'trucks': 10, 'airplanes': 20, 'ships': 30}
# Define binary and continuous variables for each transportation mode
x = {mode: model.addVar(vtype=COPT.BINARY, name=f"x_{mode}") for mode in costs}
y = {mode: model.addVar(vtype=COPT.CONTINUOUS, lb=0, name=f"y_{mode}") for mode in costs}
# Set the objective function to minimize the total transportation cost
model.setObjective(sum(costs[mode] * y[mode] for mode in costs), sense=COPT.MINIMIZE)
# Add constraints
model.addConstr(x['trucks'] + x['airplanes'] + x['ships'] >= 1, name="AtLeastOneMode")
for mode in costs:
    model.addConstr(y[mode] <= capacities[mode] * x[mode], name=f"Capacity_{mode}")
model.addConstr(x['trucks'] + x['ships'] <= 1, name="ModeExclusivity")
model.addConstr(sum(y[mode] for mode in capacities) >= 25, name="Volume Requirement")
# Solve the model
model.solve()
# Check the solution status and print the optimal values of the variables
if model.status == COPT.OPTIMAL:
    print("Optimal solution found:")
    for mode in costs:
        print(f"{mode}: x = {x[mode].x}, y = {y[mode].x}")
```

Symbolic LLM agent solvers

Carnegie Mellon University

Limitations in Existing ML4CO Solvers

Problems:

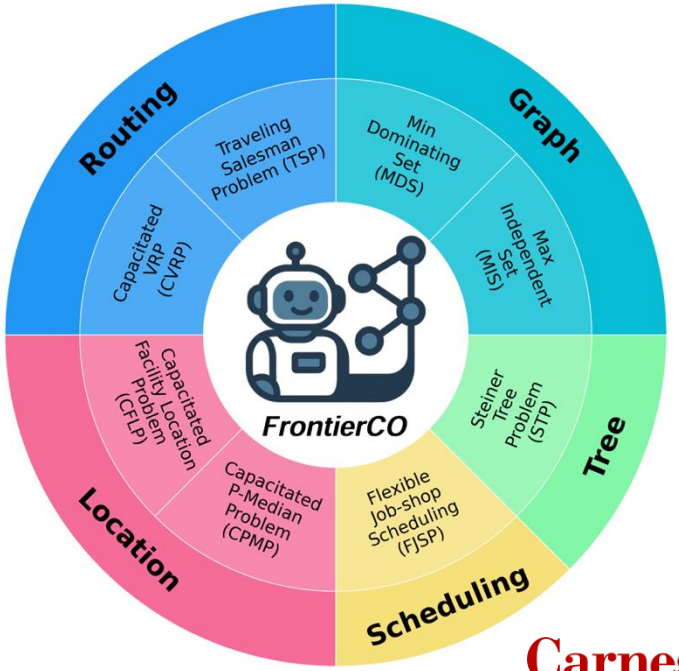
- Methods are often **heavily tuned for their own performance**
- Baselines may not receive the same level of **engineering effort**
- Reported gains can reflect **effort imbalance**, not true algorithmic improvement

FrontierCO addresses this:

- All solvers run under **identical computing resources and runtime**
- Training instances and are standardized
- Evaluation reflects **algorithmic robustness**, not manual tuning effort

FrontierCO: Real-World & Large-Scale Benchmark

- Real-world structure and extreme scale
- 8 problems from 5 domains
- Training data for each problem
- Unified protocol for classical, neural, LLM agent solvers



FrontierCO: Real-World & Large-Scale Benchmark

Table 1: Summary of collected problem instances.

Problem	Test Set Sources	Attributes	Easy Set	Hard Set
MIS	2nd DIMACS Challenge BHOSLib	Instances Nodes	36 1,404–7,995,464	16 1,150–4,000
MDS	PACE Challenge 2025	Instances Nodes	20 2,671–675,952	20 1,053,686–4,298,062
TSP	TSPLib 8th DIMACS Challenge	Instances Cities	29 1,002–18,512	19 10,000–10,000,000
CVRP	Golden et al. (1998) Arnold et al. (2019)	Instances Cities	20 200–483	10 3,000–30,000
CFLP	Avella & Boccia (2009) Avella et al. (2009)	Instances Facilities Customers	20 1,000 1,000	30 2,000 2,000
CPMP	Lorena & Senne (2004; 2000) Stefanello et al. (2015) Gnägi & Baumann (2021)	Instances Facilities Medians	31 100–4,461 10–1,000	12 10,510–498,378 100–2,000
FJSP	Behnke & Geiger (2012) Naderi & Roshanaei (2021)	Instances Jobs Machines	60 10–100 10–20	20 10–100 20–60
STP	Leitner et al. (2014) Rosseti et al. (2001)	Instances Nodes	23 7,565–71,184	50 64–4,096

Carnegie Mellon University

Benchmark Design Principle

- **Real-world diversity:** instances from TSPLib, DIMACS, etc.
- **Two difficulty levels:** easy (solvable) vs. hard (open/challenging)
- **Standardized training sets:** training instances from the same distribution
- **Unified metric:** normalized gap to best-known solution (BKS)

Representative Solvers Evaluated

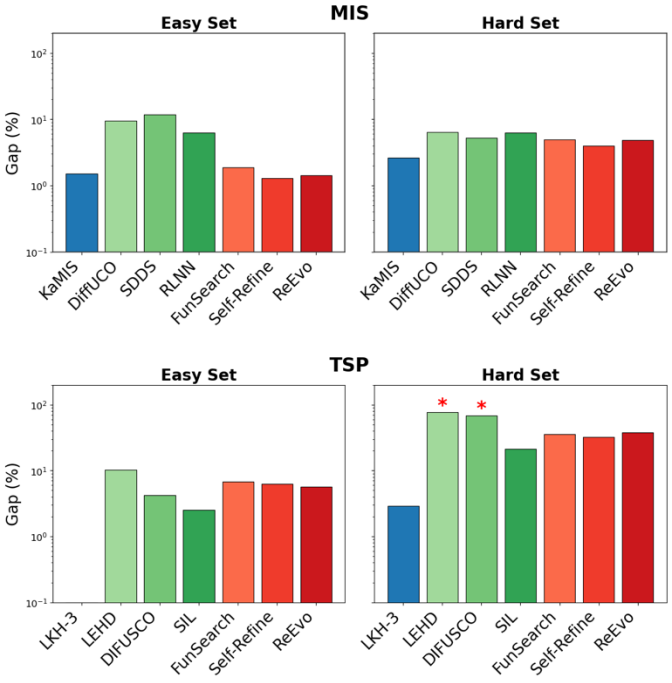
- 23 approaches across three paradigms
  - **Classical Solvers:** LKH-3, Gurobi, …
  - **Neural Models:** DIMES, DIFUSCO, RLNN, …
  - **LLM Agents:** FunSearch, ReEvo, …

- Each solver limited to 1 CPU core + 1 GPU + 1 hr runtime per instance

A truly level playing field for learning vs. algorithm reasoning

Overall Findings

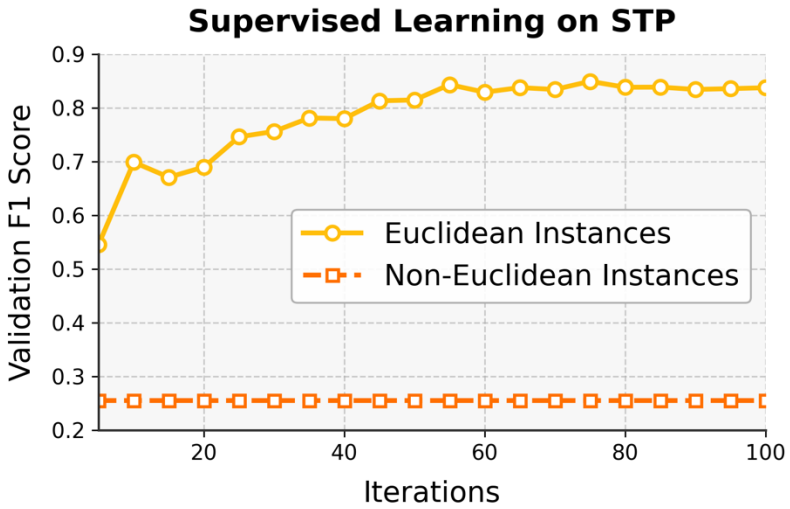
- Classical solvers remain strongest on **hard instances**
- **Neural solvers** show potential but struggle with scalability and distribution shift
- **LLM agent solvers** perform well with adaptive configuration



Carnegie Mellon University

Neural Solver Struggle with Global Constraints

- Neural solvers often fail to fully capture **global constraints**
- Instead, they rely on **local structural patterns** as heuristics
- When such structure is absent (Euclidean → non-Euclidean), performance **degrades significantly**



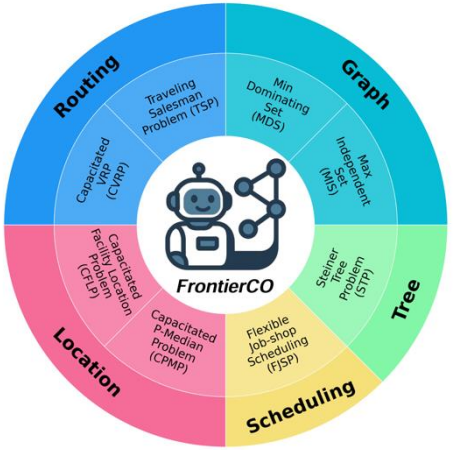
Carnegie Mellon University

LLM Agents Primarily Configure Existing Algorithms

- LLM agents achieve strong performance, but rarely **discover new algorithms**
- They mainly leverage existing heuristics (e.g., large neighborhood search, simulated annealing)
- Gains largely come from **better configuration and orchestration**

Carnegie Mellon University

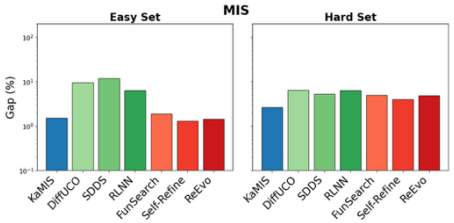
Takeaways

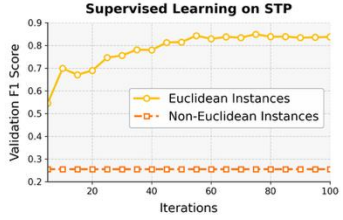


FrontierCO:

- Enables **fair comparison** across paradigms
- Removes **human tuning bias**
- Evaluates **robustness under real-world settings**

A step toward **scientific, reproducible evaluation** in ML for combinatorial optimization





Carnegie Mellon University

Carnegie Mellon University

FrontierCO: Real-World and Large-Scale Evaluation of Machine Learning Solvers for Combinatorial Optimization

Paper

Code